16. Mandatory Access Control

🛡️ Mandatory Access Control (MAC) in Linux

1. **What is MAC in Linux?**

- **MAC (Mandatory Access Control)** is a security model that enforces access control policies **centrally and mandatorily**, regardless of user discretion.
- Unlike **DAC (Discretionary Access Control)** where owners control permissions, MAC restricts all access decisions based on system-wide policies.
- In Linux, MAC is implemented mainly through frameworks like **SELinux** and **AppArmor**.

2. **How Does SELinux Enforce MAC?**

- SELinux implements MAC by assigning **security contexts (labels)** to all files, processes, and system objects.
- Access decisions are based on **policies** that define allowed interactions between these contexts.
- The kernel checks these policies on every access request and enforces **deny by default** unless explicitly allowed.
- This granular control restricts even root user actions if policy forbids them.

3. **Differences Between SELinux and AppArmor**

Feature	SELinux	AppArmor
Approach	Label-based MAC, uses **security contexts** assigned to all system objects	Path-based MAC, policies applied to file paths
Complexity	More complex, steeper learning curve	Simpler, easier to configure
Policy granularity	Fine-grained, detailed policy control	Coarser control, profile-based
Enforcement style	Default-deny, denies unless allowed	Default-allow, restricts based on profile
Use cases	Enterprise, government, high-security systems	Desktop and server environments needing simpler control

4. **What is the Purpose of Policy in MAC Systems?**

- A **policy** defines the **rules and constraints** that govern access to system resources.
- Policies specify **who (subject)** can do **what (action)** on **which resource (object)**.
- Ensures system-wide consistent enforcement of security rules.

- Policies enable **least privilege** by limiting access only to necessary resources.

---

5. How Do Labels Work in SELinux?

- Every object (file, process, socket, etc.) is assigned a **security context label**, typically formatted as:

  ```
  user:role:type:level
  ```

- **User:** SELinux user identity
- **Role:** Role-based access control grouping (e.g., sysadm_r)
- **Type:** Primary classification used in Type Enforcement (most important)
- **Level:** MLS (Multi-Level Security) sensitivity level (optional)
- Access decisions are made based on these labels matching the policy rules.

---

6. What Are Type Enforcement (TE), Role-Based Access Control (RBAC), and Multi-Level Security (MLS) in SELinux?

- **Type Enforcement (TE):** Core SELinux mechanism controlling access between **types** (labels). Defines allowed interactions between processes and objects.
- **Role-Based Access Control (RBAC):** Controls what roles users can assume, limiting the types and permissions accessible to those roles.
- **Multi-Level Security (MLS):** Adds sensitivity levels to control access based on classification (e.g., Confidential, Secret), often used in government/military contexts.

---

7. How Can You Check the Status of SELinux on a System?

- Use:

  ```
  sestatus
  ```

- Or:

  ```
  getenforce
  ```

  - Outputs: `Enforcing`, `Permissive`, or `Disabled`.

- Or check config file:

  ```
  cat /etc/selinux/config
  ```

---

8. Common SELinux Management Commands

- `getenforce` – Show current mode (Enforcing, Permissive, Disabled).
- `setenforce [Enforcing|Permissive]` – Temporarily change mode.
- `sestatus` – Detailed status.
- `semanage` – Manage SELinux policies (labels, ports, booleans).

- `restorecon` – Restore default contexts on files/directories.
- `chcon` – Change context of a file temporarily.
- `audit2allow` – Generate SELinux policy allow rules from audit logs.

---

9. How Do You Set File Contexts in SELinux?

- Use `semanage fcontext` to add or modify file context rules permanently:

```
semanage fcontext -a -t httpd_sys_content_t "/myweb(/.*)?"
```

- Apply contexts with:

```
restorecon -Rv /myweb
```

- Temporary change:

```
chcon -t httpd_sys_content_t /myweb/index.html
```

---

10. What is an AppArmor Profile?

- A **profile** in AppArmor is a set of rules applied to an application or process, specifying allowed file accesses, capabilities, and network actions.
- Profiles are **path-based** rather than label-based.
- Profiles can be in **complain mode** (logging violations but allowing actions) or **enforce mode** (blocking violations).

---

11. How Do You Reload AppArmor Profiles?

- Reload all profiles:

```
sudo systemctl reload apparmor
```

- Or individually:

```
sudo apparmor_parser -r /etc/apparmor.d/profile_name
```

- Check status with:

```
sudo aa-status
```

---

12. What is the Concept of Least Privilege in MAC?

- **Least privilege** means giving users/processes **only the permissions necessary** to perform their function, and no more.
- MAC systems enforce this at kernel level, limiting damage scope if an application or user is compromised.

---

13. How Do You Troubleshoot SELinux Issues?

- Check **audit logs** for denied actions:

  ```
  ausearch -m AVC,USER_AVC -ts recent
  ```

- Convert audit denials to allow rules:

  ```
  audit2allow -w -a
  audit2allow -a -M mypol
  semodule -i mypol.pp
  ```

- Temporarily set SELinux to **permissive** mode to diagnose:

  ```
  setenforce 0
  ```

- Use `sealert` tool for GUI help (on desktops).
- Review contexts and restore if mislabeled.

---

14. Significance of Audit Logs in MAC Systems

- Audit logs capture **policy denials and enforcement actions** in MAC systems.
- Critical for diagnosing access issues and security incidents.
- Helps refine and tune security policies.
- Usually found in `/var/log/audit/audit.log`.

---

15. Explain the Concept of Capabilities in Linux Security

- Linux **capabilities** break root privileges into smaller units that can be independently assigned.
- Allows processes to have specific elevated rights without full root privileges.
- Examples:
  - `CAP_NET_BIND_SERVICE` allows binding to low-numbered ports (<1024).
  - `CAP_SYS_ADMIN` is very powerful, akin to root access.
- Managed via `setcap` command.

---

16. How to Use `semanage`

- `semanage` is a tool to manage SELinux policy components (file contexts, ports, booleans, users, etc).
- Examples:
  - Add file context:

    ```
    semanage fcontext -a -t httpd_sys_content_t "/var/www/html(/.*)?"
    ```

  - Add port context:

    ```
    semanage port -a -t http_port_t -p tcp 8080
    ```

- Modify booleans (toggle settings):

  ```
  setsebool -P httpd_can_network_connect on
  ```
